

1 **METHOD AND APPARATUS FOR SERVING CONTENT FROM A**
2 **SEMI-TRUSTED SERVER**

3 **FIELD OF THE INVENTION**

4 This invention is directed to the field of computer
5 networks. It is more particularly directed to the
6 Internet and specifically to web-servers which store
7 and transmit information to their clients.

8 **BACKGROUND OF THE INVENTION**

9 Much of the communication in the Internet involves at
10 least one client making a request to a web-server, and
11 the web-server responding to the client's request. By
12 web-server, we mean a device or collection of devices
13 (e.g., datastore, directory, machines, and software)
14 that communicates with a client using the HTTP
15 Protocol. For a particular web application, we define
16 an origin web-server to be a web-server that is
17 completely trusted with the functions and data of a web
18 application with regard to the security policy for the
19 application.

20 As a way to shorten the length of time that the client
21 must wait for a response and to lighten the load on the
22 Internet and origin web-server, techniques have been
23 developed to allow the client to be serviced by a proxy
24 web-server, also referred to simply as a proxy, where

1 the proxy web-server is usually closer to the client or
2 more lightly loaded than the origin web-server.

3 Proxy web-servers can be integrated into the Internet
4 communication process in several different ways. In
5 some configurations, clients always make requests to a
6 proxy web-server rather than the origin web-server.
7 The proxy web-server may respond to the client,
8 fetching content from the origin web-server as
9 necessary, or the proxy web-server may refer the client
10 to another proxy web-server or the origin server if the
11 proxy web-server is unable to satisfy the client's
12 request. In other configurations, a client first makes
13 a request to the origin web-server. The origin
14 web-server may refer the client to a proxy for the
15 current request or all future requests, or the origin
16 web-server may respond to part of the client's request,
17 but refer the client to a semi-trusted web-server for a
18 portion of the response.

19 In most cases, the content offloaded to a proxy
20 web-server has been limited to non-sensitive data, so
21 that access control schemes are not required.
22 Non-sensitive data is defined as data which does not
23 require any access control, and may be accessible to
24 any user on the network. On a typical web-page,
25 embedded images are an example of non-sensitive data.
26 On the other hand, restricted data or sensitive data is
27 defined as data which has some restrictions on who can
28 obtain it. Examples of restricted data include pages
29 that are obtained by subscription to a set of
30 registered users, images that are available to a

1 restricted set of users, or data can is personalized
2 for a specific user.

3 Common subscription services and personalized content
4 on the Internet are increasing, and they should also
5 benefit from the performance gains afforded by proxy
6 web-servers. The restricted information requires the
7 proxy web-servers to have access control methods in
8 place, but the situation is complicated because in many
9 cases the proxy web-servers are not under the control
10 of the content providers. Such proxy web-servers fall
11 into the class of semi-trusted web-servers. For a
12 particular web application, we define a semi-trusted
13 web-server to be a web-server that is partially trusted
14 for the functions of the application with regard to the
15 security policy for the application. In particular, a
16 semi-trusted web-server may be trusted for
17 authorization, access to user identifiers, SSL
18 tunneling to an origin web-server for content, and
19 non-sensitive transactions, but the semi-trusted
20 web-server may not be trusted with long-term sensitive
21 data such as user passwords or secret keys for an
22 origin web-server.

23 **SUMMARY OF THE INVENTION**

24 It is therefore an aspect of the present invention to
25 provide a method to enable a client to access
26 restricted information from an origin web-server
27 through a semi-trusted web-server.

1 It is a further aspect of the invention to provide an
2 apparatus to enable a client to access restricted
3 information from an origin web-server through a
4 semi-trusted web-server.

5 It is a further aspect of the invention to reduce the
6 security risk of using cookies for authentication.

7 It is a further aspect of the invention to protect data
8 stored at a semi-trusted web-server.

9 In an example of the invention, a client would use the
10 method disclosed herein to gain access to restricted
11 information at a semi-trusted web-server.

12 **BRIEF DESCRIPTION OF THE DRAWINGS**

13 These and other aspects, features, and advantages of
14 the present invention will become apparent upon further
15 consideration of the following detailed description of
16 the invention when read in conjunction with the drawing
17 figures, in which:

18 Fig. 1 shows an example of an environment having a
19 client, an origin web-server, and a semi-trusted
20 web-server, and the problems associated with a client
21 accessing restricted information through the
22 semi-trusted web-server.

23 Fig. 2 shows the deployment of the method described in
24 this invention for enabling a client to access
25 restricted information in the environment described by

1 Fig. 1. It shows the authenticator, creator,
2 presentator, and correlator.

3 Fig. 3 shows a first embodiment of this invention
4 illustrating a technique for presenting a client
5 credential for the case when the semi-trusted
6 web-server and the origin web-server are located in the
7 same name space domain, (e.g. proxy.company.com and
8 company.com respectively).

9 Fig. 4 shows a flowchart that illustrates the actions
10 taken by the origin web-server for the embodiment shown
11 in Fig. 3.

12 Fig. 5 shows a flowchart that illustrates the actions
13 taken by the semi-trusted web-server for the embodiment
14 shown in Fig. 3.

15 Fig. 6 shows a second embodiment in which the client
16 credential is presented to the semi-trusted web-server
17 by the origin web-server and the origin server
18 redirects the client to the semi-trusted web-server
19 with HTTP redirection. This embodiment addresses the
20 case when the origin web-server and semi-trusted
21 web-server are not located in the same domain.

22 Fig. 7 shows a third embodiment in which correlation is
23 aided by the origin web-server installing a first
24 client-side program in the client browser that sends
25 client-specific information to the origin web-server
26 and by the semi-trusted web-server installing a second
27 client-side program in the client browser that sends

1 client-specific information to the semi-trusted
2 web-server.

3 Fig. 8 shows the sequence of events for the embodiment
4 shown in Fig. 7.

5 Fig. 9 shows a flowchart that illustrates the actions
6 taken by the origin web-server for the embodiment shown
7 in Fig. 7.

8 Fig. 10 shows a flowchart that illustrates the actions
9 taken by the semi-trusted web-server for the embodiment
10 shown in Fig. 7.

11 Fig. 11 shows a flowchart that illustrates the actions
12 taken by the origin web-server for creating a cookie
13 that will be presented to the semi-trusted web-server.

14 Fig. 12 shows a flowchart that illustrates the actions
15 taken by the origin or semi-trusted web-server when
16 validating a cookie as part of the correlation
17 procedures.

18 Fig. 13 shows a flowchart that illustrates the actions
19 taken by the origin or semi-trusted web-server when
20 updating a cookie as part of the procedures described
21 above.

22 Fig. 14 illustrates the use of shared keys by
23 semi-trusted and origin web-servers.

1 Fig. 15 shows a flowchart that illustrates the actions
2 taken by the origin web-server when creating secure
3 content for distribution through a semi-trusted
4 web-server.

5 Fig. 16 shows a flowchart that illustrates the actions
6 taken by the semi-trusted web-server when it serves
7 partly secure content.

8 Fig. 17 shows a flowchart that illustrates the actions
9 taken by the client when receiving secure content.

10 Fig. 18 illustrates the use of a new content-type that
11 distinguishes secure content.

12 Other objects and a better understanding of the
13 invention may be realized by referring to the detailed
14 description.

15 DESCRIPTION OF THE INVENTION

16 The present invention enables a client to access
17 restricted information through a semi-trusted
18 web-server. A typical environment in which the access
19 occurs is illustrated in Fig. 1. The figure shows a
20 client, an origin web-server, and a semi-trusted
21 web-server which are connected to a core network such
22 as the Internet.

23 The client 101, semi-trusted web-server 104, and origin
24 web-server 103 are connected to a core network 102.

1 The client, semi-trusted web-server, and origin
2 web-server can be directly connected to the core
3 network as exemplified in the figure, or they can be
4 connected via intermediary firewalls, routers, and
5 subnetworks. The semi-trusted web-server may be on the
6 route that packets in the network follow when the
7 client communicates with the origin web-server.

8 To shorten the length of time that the client must wait
9 for a response to requests and to lighten the load on
10 the origin web-server, the semi-trusted web-server may
11 service the requests of the client. Generally, a
12 semi-trusted web-server is chosen to service the
13 requests of a client if the semi-trusted web-server is
14 closer to the client than the origin web-server or if
15 the semi-trusted web-server is less loaded than the
16 origin web-server. In some configurations, clients
17 always make requests to a semi-trusted web-server
18 rather than the origin web-server. The semi-trusted
19 web-server may respond to the client, fetching content
20 from the origin web-server as necessary, or the
21 semi-trusted web-server may refer the client to another
22 semi-trusted web-server or the origin server if the
23 semi-trusted web-server is unable to satisfy the
24 client's request. In other configurations, a client
25 first makes a request to the origin web-server. The
26 origin web-server may refer the client to a
27 semi-trusted web-server for the current request or all
28 future requests, or the origin web-server may respond
29 to part of the client's request but refer the client to
30 a semi-trusted web-server for a portion of the
31 response.

1 In Fig. 1, the semi-trusted web-server is not
2 necessarily considered to be under the control of the
3 content provider or the origin web-server such that the
4 semi-trusted web-server is not trusted with regard to
5 the security policy for long-term sensitive data.
6 However, the semi-trusted web-server is trusted for
7 authorization, user identifiers, non-sensitive
8 transactions, and SSL tunneling for content. Because
9 the semi-trusted web-server is semi-trusted by the
10 origin web-server, access control methods are needed to
11 protect the restricted information at the semi-trusted
12 web-server and to protect sensitive user credentials.
13 Our patent provides techniques to provide access
14 control to restricted information at the semi-trusted
15 web-server without risking long-term sensitive data at
16 the semi-trusted web-server such as user passwords.

17 To enable a client to access restricted information at
18 the semi-trusted web-server without risking long-term
19 sensitive data, additional functions are needed. Fig.
20 2 shows the structure of any machine that is running in
21 the network that can deploy the invention disclosed
22 herein. The machine 201 may be distributed among the
23 client, semi-trusted web-server, origin web-server,
24 trusted agents of the client, trusted agents of the
25 semi-trusted web-server, and trusted agents of the
26 origin web-server. Any device 201 implementing this
27 invention consists of four components including an
28 authenticator 202, credential creator 203, credential
29 presentator 204, and credential correlator 205. In
30 some embodiments of this invention, the authenticator

1 202 may be provided by external services which are
2 already present in the network. The authenticator 202
3 first authenticates the client 101 to the origin
4 web-server 103. The credential creator 203 generates
5 credential for the client to be used for subsequent
6 communications. The credential presenter 204
7 communicates the client credential to the semi-trusted
8 web-server 104. The credential correlator 205
9 correlates the client credential with the accessing
10 client and the client user identifier.

11 When a client requests restricted information, the
12 client is referred to the origin web-server for
13 authentication. After credential creation, the client
14 is referred back to the semi-trusted web-server to
15 repeat the request and the credential presented to the
16 semi-trusted web-server. After the credential
17 correlation, the semi-trusted web-server responds to
18 the client's request by providing the requested content
19 if the credential is valid.

20 Fig. 3 shows a first embodiment of this invention
21 illustrating a technique for presenting a client
22 credential for the case when the semi-trusted
23 web-server 300 and the origin web-server 310 are
24 located in the same name space domain (e.g.
25 proxy.company.com and company.com respectively). In
26 this embodiment, the client makes a request to the
27 origin web-server. The request is sometimes referred
28 to the origin web-server by the semi-trusted
29 web-server. The client is authenticated to the origin
30 web-server with the authentication 320 component. At

1 the origin web-server, the credential creation 325
2 component creates a client credential in the form of an
3 HTTP cookie which includes client-specific environment
4 information such as the apparent client Internet
5 Protocol (IP) address and the HTTP header information.
6 The origin web-server sets the cookie on the client and
7 the client is referred to the semi-trusted web-server
8 by the origin web-server using HTTP redirection.
9 Because the semi-trusted web-server and origin
10 web-server are in the same name-space domain, the
11 credential cookie will be presented to the semi-trusted
12 web-server 315 when the client makes a request to the
13 semi-trusted web-server. At the semi-trusted
14 web-server, the credential correlation 330 component
15 correlates the presented cookie to the client-specific
16 environment information such as the apparent client IP
17 address and the HTTP header information. If the cookie
18 is valid, the semi-trusted web-server responds to the
19 request using the access control of the user identifier
20 specified in the cookie. The format and validation of
21 the cookie are illustrated in Fig. 11.

22 Fig. 4 shows a flowchart that illustrates the actions
23 taken by the origin web-server 310 for the embodiment
24 shown in Fig. 3. The flowchart is entered in the step
25 401 whenever the device implementing the embodiment is
26 started at the origin web-server 310. In step 405,
27 the origin web-server waits for messages from a client.
28 Upon receiving a message, in step 410 the origin
29 web-server checks to see if the client is
30 authenticated. The client may be authenticated by
31 presenting a valid client credential as illustrated in

1 Fig. 12, or the client may be authenticated by another
2 scheme. If the client is not authenticated in step 410,
3 then in step 425 the origin web-server initiates
4 authentication and returns to step 410. If the client
5 is authenticated in step 410, then step 415 is
6 executed. In step 415, the origin web-server uses
7 client-specific environment information to create a
8 client credential which is stored in a cookie that is
9 set on the client. Next, in step 420, the origin
10 web-server refers the client to a semi-trusted
11 web-server. Finally, the origin web-server returns to
12 step 405 and waits for another message.

13 Fig. 5 shows a flowchart that illustrates the actions
14 taken by the semi-trusted web-server 300 for the
15 embodiment shown in Fig. 3. The flowchart is entered
16 in step 500 whenever the device implementing the
17 embodiment is started at the semi-trusted web-server
18 300. In step 505, the semi-trusted web-server waits
19 for messages from clients. Upon receiving a message,
20 the semi-trusted web-server checks to see if the client
21 has submitted a cookie containing a valid client
22 credential in step 510. Step 510 involves correlating
23 the client credential in the cookie with
24 client-specific information that the semi-trusted
25 web-server obtains from the client as illustrated in
26 Fig. 12. If the client credential is not valid in step
27 510, then step 530 is executed. In step 530, the
28 semi-trusted web-server refers the client to the origin
29 web-server and returns to step 505. If the client
30 credential is valid in step 510, then step 515 is
31 executed. In step 515, the cookie and client

1 credential is updated as illustrated in Fig. 13. After
2 step 515, the semi-trusted web-server checks to see if
3 the client is authorized in step 520. If the client is
4 not authorized in step 520, then step 535 is executed.
5 In step 535, the semi-trusted web-server sends a
6 forbidden message to the client and returns to step
7 505. If the client is authorized in step 520, then
8 step 525 is executed and the content is provided to the
9 client. Following step 525, the semi-trusted
10 web-server returns to step 505 and waits for more
11 messages.

12 Fig. 6 shows a second embodiment in which the client
13 credential is presented to the semi-trusted web-server
14 600 by the origin web-server 610 and the origin server
15 610 redirects the client to the semi-trusted web-server
16 635 with HTTP redirection. This embodiment addresses
17 the case when the origin web-server and semi-trusted
18 web-server are not located in the same domain, although
19 it can also be used if the servers are in the same
20 domain. In this embodiment, the client makes a request
21 to the origin web-server. The request is sometimes
22 referred to the origin web-server by the semi-trusted
23 web-server. The client is authenticated to the origin
24 web-server with the authentication 630 component. At
25 the origin web-server, the credential creation 620
26 component creates a client credential in the form of an
27 HTTP cookie which includes client-specific environment
28 information such as the apparent client Internet
29 Protocol (IP) address and the HTTP header information.
30 The presentation component 625 sends the client
31 credential to the semi-trusted web-server 635, and then

1 origin web-server 610 refers the client to the
2 semi-trusted web-server by HTTP redirection including a
3 reference to the client credential cookie in the HTTP
4 redirection URL. When the client 605 makes a request
5 to the semi-trusted web-server 635, the presentation
6 component 615 gives a reference to the client
7 credential cookie in the form of a URL query string, or
8 if the client already has a client credential cookie
9 the cookie is presented to the semi-trusted web-server.
10 In the correlation component 635, the reference to the
11 client credential cookie is used to select one of the
12 stored cookies at the semi-trusted web-server, and then
13 client-specific environment information such as the
14 apparent client IP address and the HTTP header
15 information is correlated to the client credential as
16 illustrated in Fig. 12. If the client matches the
17 client credential, then the semi-trusted web-server
18 sets the cookie on the client for future requests and
19 the semi-trusted web-server responds to the request
20 using the access control of the user identifier
21 specified in the cookie. The format and validation of
22 the cookie are illustrated in Fig. 11.

23 Fig. 7 shows a third embodiment in which correlation is
24 aided by the origin web-server installing a first
25 client-side program in the client browser that sends
26 client-specific information to the origin web-server
27 and by the semi-trusted web-server installing a second
28 client-side program in the client browser that sends
29 client-specific information to the semi-trusted
30 web-server. This embodiment addresses the case when
31 the origin web-server and semi-trusted web-server are

1 not located in the same domain, although it can also be
2 used if the servers are in the same domain. This
3 embodiment further reduces the risk of using cookies
4 for authentication by using more client-specific
5 environment information. In this embodiment, the
6 client makes a request to the origin web-server
7 (possibly being referred to the origin web-server by
8 the semi-trusted web-server). The client is
9 authenticated to the origin web-server with the
10 authentication 730 component, which may be enhanced
11 with the first client-side program. At the origin
12 web-server, the credential creation 720 component
13 creates a client credential in the form of an HTTP
14 cookie which includes client-specific environment
15 information such as the apparent client Internet
16 Protocol (IP) address and the HTTP header information,
17 as well as additional client-specific environment
18 information that is collected by the first client-side
19 program such as a hash of the local IP address and
20 browser application process identifier. The
21 presentation component 725 sends the client credential
22 to the semi-trusted web-server 735, and then origin
23 web-server 710 refers the client to the semi-trusted
24 web-server by HTTP redirection including a reference to
25 the client credential cookie in the HTTP redirection
26 URL. When the client 705 makes a request to the
27 semi-trusted web-server 735, the presentation component
28 715 gives a reference to the client credential cookie
29 in the form of a URL query string, or if the client
30 already has a client credential cookie the cookie is
31 presented to the semi-trusted web-server. In the
32 correlation component 735, the reference to the client

1 credential cookie is used to select one of the stored
2 cookies at the semi-trusted web-server, and then
3 client-specific environment information such as the
4 apparent client IP address and the HTTP header
5 information as well as additional client-specific
6 environment information obtained by the correlation
7 component 740 through the second client-side program is
8 correlated to the client credential. If the client
9 matches the client credential, then the semi-trusted
10 web-server sets the cookie on the client for future
11 requests and the semi-trusted web-server responds to
12 the request using the access control of the user
13 identifier specified in the cookie. The format and
14 validation of the cookie are illustrated in Fig. 11.

15 Fig. 8 shows the message exchange that occurs in the
16 third embodiment as illustrated in Fig. 7. A client
17 sends an HTTP request (800) to the origin web-server.
18 The origin web-server replies with an authentication
19 request message (805), which is presented to the user
20 as a user ID and password prompt. The client replies
21 with an authentication response message (810) including
22 the authentication data. If the user ID and password
23 are validated successfully at the origin web-server
24 then the origin web-server sends a client program to
25 the client (815). The client program runs on the client
26 machine and gathers local environment information,
27 e.g., process ID and user ID of the client's browser
28 and local IP address. This information is hashed in the
29 client for privacy and sent back as client specific
30 correlation information to the origin web-server (820).
31 The origin web-server creates a valid cookie as

1 described later in Fig. 11 and includes a reference to
2 the cookie in the redirection of the client (825). The
3 original web-server also stores the cookie at the
4 semi-trusted web-servers (830) or in a directory
5 accessible to the semi-trusted web-servers. The client
6 proceeds with sending the request to the semi-trusted
7 web-server (835) as indicated in the redirection. If
8 the semi-trusted web-server and original web-server
9 share one domain, then the client includes the cookie.
10 If the client does not include a cookie, then the
11 semi-trusted web-server looks up the cookie from where
12 it is stored by the origin web-server. If the
13 semi-trusted web-server does not receive a cookie from
14 the client and does not find a cookie in its accessible
15 storage, then the semi-trusted web-server redirects the
16 client to the original web-server and the process
17 restarts at message 800. The semi-trusted web-server
18 can implement protection against oscillating
19 redirections by allowing a client only a small number
20 of redirections in a specified time frame. If the
21 semi-trusted web-server has access to the client's
22 cookie, then it sends a correlation client program to
23 the client (840), which gathers client environment
24 information (see 815) and send a hash of it back to the
25 proxy web-server (845). The semi-trusted web-server
26 then verifies whether the client's environment hash
27 matches the environment hash stored in the cookie. If
28 not, then the semi-trusted web-server redirects the
29 client to the origin web-server and the scenario
30 restarts (800). If it matches, then the semi-trusted
31 web-server validates the cookie (as described later in
32 Fig. 11). If the cookie is valid, then the semi-trusted

1 web-server extracts the client's credentials from the
2 cookie and continues with the authorization. If the
3 client's credentials are sufficient to access the
4 requested information then access is provided to the
5 client, if not then the access is denied (850). The
6 semi-trusted web-server updates the cookie (as
7 described in Fig. 11), sends the updated cookie with
8 message 850 to the client, and stores the updated
9 cookie in the accessible storage for semi-trusted
10 web-servers.

11 Fig. 9 shows a flowchart that describes the behavior of
12 the origin web-server in the third embodiment as shown
13 in Fig. 7. After starting the origin web-server (900),
14 it waits for request messages (905). When receiving a
15 client request, the origin web-server gathers client
16 environment information (910) as described in Fig. 8 by
17 sending a client side program to the client, which
18 reports respective information. The origin web-server
19 examines whether a valid cookie for this client exists
20 (915). If yes, then the client is referred to a proxy
21 web-server (920) and the origin web-server waits for
22 other requests (905). Otherwise, the origin web-server
23 authenticates the client (925) by sending an
24 authentication request to the client and waiting for
25 the authentication response (as described in Fig. 8,
26 messages 805 and 810). If the authentication fails,
27 then an access forbidden message is sent to the client
28 (835) and the origin web-server waits for other
29 requests (905). In this case, the origin web-server
30 should audit the number of failed authentication
31 procedures per client and lock accounts where a certain

1 limit of successive authentication requests have
2 failed. If the authentication is successful and no
3 cookie is present then the origin web-server creates a
4 cookie for the client as shown in Fig. 11, and sends
5 the cookie to the client together with a redirection
6 message that refers the client to a semi-trusted
7 web-server (940). If origin and semi-trusted
8 web-servers do not share the same domain, the cookie
9 must be stored in a directory accessible to the
10 semi-trusted web-servers. Otherwise this is optional.
11 The origin web-server then returns to the state where
12 it waits for incoming requests (905).

13 Fig. 10 shows a flowchart that describes the behavior
14 of the semi-trusted web-server in the third embodiment
15 as shown in Fig. 7. After starting the semi-trusted
16 web-server (1000), it waits for incoming requests
17 (1005). Next, the semi-trusted web-server checks
18 whether it has as cookie for this client (1010). If the
19 client does not submit a valid cookie with the request
20 and the semi-trusted web-server does not have the
21 client's cookie in other storage, e.g., in a directory,
22 then the client is referred to the origin web-server
23 (1015) for cookie creation. Otherwise, the semi-trusted
24 web-server gathers client environment information
25 (1020) using a client side program as described in Fig.
26 7. The semi-trusted web-server then verifies whether
27 the gathered information and the client information
28 stored in the cookie match (1025) as illustrated in
29 Fig. 12. If it does not, then the client is referred to
30 the origin web-server to create a valid cookie (1015).
31 If it does, then the semi-trusted web-server updates

1 the cookie (1035) as described in Fig. 13. Then the
2 semi-trusted web-server extracts the client's
3 credentials from the cookie and checks the
4 authorization of the client to access the requested
5 information (1040). If the client is not authorized to
6 access the requested information then the semi-trusted
7 web-server sends a forbidden message to the client
8 (1045) and returns to the waiting state (1005). If the
9 client is authorized then the client is provided access
10 (1050). Afterwards, the semi-trusted web-server returns
11 to the waiting state (1005).

12 Fig. 11 illustrates an example of a client cookie that
13 is created by the origin web-server. The cookie
14 consists of two parts: one part is encrypted and one
15 part is not encrypted. The encrypted part consists of
16 the client's IP address as seen by the origin
17 web-server (1100); optional client correlation
18 information gathered either by a client-side program
19 (1105); a hash of the client's request header as seen
20 by the origin web-server (1110); the client's user
21 identification as used for authorization by the origin
22 and semi-trusted web-server (1115); optionally a random
23 bit pattern B (1120); a time stamp including the
24 creation time of the cookie (1125); a global time out
25 value valid for the whole domain (1130) which is
26 usually a fixed offset added to the creation time; and
27 a cookie inactivity time-out (1135) which is a fixed
28 offset added to the cookie creation time. The clear
29 part of the cookie includes the key identification
30 number (1140), which denotes the key used to encrypt
31 the upper part of the cookie; the domain name of the

1 origin web-server (1145); optionally a copy of the
2 encrypted random bit pattern B (1150); and finally a
3 digital signature (1155) over the fields above whereby
4 all fields are used in the clear for creating the
5 signature. Afterwards, the fields of the first part are
6 encrypted using the key Kc (1160) shared by the
7 semi-trusted web-server and the origin web-server.

8 Fig. 12 shows a flowchart illustrating the process of
9 validating a client cookie and returning client
10 credentials in case the cookie is valid as part of the
11 correlation procedures. The client cookie may be
12 present in the HTTP request by the client, or the
13 client may provide a reference to a cookie that is
14 stored at the semi-trusted web-server or that is
15 available to the semi-trusted web-server through an
16 external device or directory. If no cookie is found,
17 then the client is referred to the origin web-server so
18 that the client can be authenticated and a cookie can
19 be created. The cookie validation procedure starts at
20 step 1200. In step 1205, the cookie is decrypted by
21 using the domain identifier and key identifier to
22 select an appropriate decryption key and by performing
23 the decrypt operation. In step 1210, the global
24 time-out and inactivity time-out fields are checked to
25 see if the cookie has expired and the bit pattern is
26 compared to the bit pattern copy. If the cookie has
27 expired with either the global time-out or the
28 inactivity time-out or if the bit pattern and bit
29 pattern copy are not equal, then the cookie is invalid
30 and step 1215 is executed. In step 1215, the process
31 returns invalid and stops. If the cookie has not

1 expired and the bit patterns match in step 1210, then
2 step 1220 is executed. In step 1220, if the key used
3 to decrypt the cookie in step 1205 has been marked as
4 compromised or if heightened security is desired, step
5 1225 is executed; otherwise step 1230 is executed. In
6 step 1225, the signature of the cookie is checked. If
7 the signature does not match, then the cookie is
8 invalid and step 1215 is executed. If the signature
9 does match, then step 1230 is executed. In step 1230,
10 client-specific environment information is gathered by
11 the semi-trusted web-server. Some client-specific
12 environment information such as apparent IP address and
13 HTTP header hash may come from the client connection,
14 while other client-specific environment information
15 such as a hash of the local user identifier, browser
16 process identifier, and IP address may be sent to the
17 semi-trusted web-server by the second client-side
18 program. After step 1230, in step 1235 the hash of
19 client-specific environment information is checked for
20 equality with the client-specific environment
21 information stored in the cookie. If there is not
22 equality in step 1235, then the cookie is invalid and
23 step 1215 is executed. If there is equality in step
24 1235, then step 1240 is executed. In step 1240, the
25 client access credentials in the cookie are retrieved.
26 In step 1245, the validation process reports valid and
27 returns the client's credentials to the caller. These
28 credentials are used throughout the authorization to
29 decide whether to provide access or not to the client.

30 Fig. 13 shows a flowchart illustrating the process of
31 updating a client cookie. After starting execution in

1 step 1300, step 1305 is executed. Step 1305 checks to
2 see whether or not the cookie has been marked valid in
3 the validation process illustrated in Fig. 12. If the
4 cookie is not valid in step 1305, then step 1330 is
5 executed. In step 1330, the cookie is deleted from the
6 client, local storage, and any cookie directory or
7 storage device and step 1325 is executed. In step
8 1325, the cookie update process stops. If the cookie is
9 valid in step 1305, then a new inactivity time-out is
10 set in step 1310 and step 1315 is executed. In step
11 1315, the semi-trusted web-server checks the key
12 management system to see if the key used by the client
13 is still valid. The key may not be valid if the key
14 has timed out or if the origin web-server has sent a
15 message to the semi-trusted web-server indicating that
16 the key is no longer valid. If in step 1315, the key
17 is no longer valid, then step 1330 is executed. If in
18 step 1315, the key is valid, then in step 1320 the
19 cookie is encrypted with the same shared key indicated
20 in the key identifier field and step 1325 is executed.

21 Fig 14 illustrates the key entry used by semi-trusted
22 web-servers and origin web-servers to protect part of
23 the cookie both against unnoticed modification and
24 disclosure. The key entry comprises of a key
25 identification number (1400), the key itself (1405),
26 and optionally a key time-out value (1410). At the
27 beginning, the origin web-server creates a key entry
28 with a new key, a new key identification number, and an
29 empty time-out value and distributes it securely to all
30 semi-trusted web-servers. The origin server includes in
31 the appropriate cookie field (see Fig. 11), the key

1 identification of the key that is used when creating
2 the cookie. Once a semi-trusted web-server is known to
3 be compromised, the origin web-server issues a new key
4 entry with a new key identification number to those
5 semi-trusted web-servers that are not compromised; this
6 new key issue triggers the semi-trusted web-servers to
7 set a global time-out value to the old key (e.g. 5
8 seconds). After this time-out value expires, cookies
9 using the old key are no longer accepted (see Fig. 13)
10 and clients presenting them are redirected to the
11 origin web-server for authentication and creation of a
12 new cookie. This way, cookies and user information
13 related to cookies that were known by the compromised
14 web server lose their value for replay attacks.

15 Fig. 15 shows a flowchart that illustrates the actions
16 taken by the origin web-server when creating secure
17 content for a client (1500). This part applies if the
18 client retrieves information that shall be kept
19 confidential even regarding the semi-trusted
20 web-servers serving these data. Embedding secure
21 content into information served by semi-trusted
22 web-servers enhances scalability and security as most
23 of the information retrieved is not highly sensitive
24 and can be shared by many users leveraging caching in
25 the semi-trusted web-server. The small amount of
26 individual and sensitive data is retrieved
27 transparently for the client and automatically by the
28 semi-trusted web-server from the origin web-server. A
29 special secure content handler, installed in the
30 client, will present the secure content to the user in
31 a way that makes secure and conventional data

1 distinguishable for the user. To enable secure content,
2 the content provider determines the key Kclient, which
3 it shares with the client (1505). It then encrypts the
4 sensitive content with a key Kcontent (1510). The key
5 Kcontent is encrypted with Kclient and added to the
6 secure content (1515). The content is marked with a new
7 tag (1520) and then served either directly to the
8 client or to the semi-trusted web-server which embeds
9 the secure content as part of the content it serves to
10 this client (1525). The procedure ends with 1530. Using
11 an independent key Kcontent to encrypt sensitive
12 content supports the use of encrypted content for
13 different users which do not share the key Kclient; the
14 origin web-server just needs to encrypt the common key
15 multiple times instead of the whole content.

16 Fig. 16 shows a flowchart that illustrates the actions
17 taken by the semi-trusted web-server when it serves
18 partly secure content (1600). If the secure content is
19 retrieved by the semi-trusted web-server, then client
20 identification information (e.g. the client's cookie)
21 is determined (1605) and presented to the origin
22 web-server when retrieving the secure content from the
23 origin web-server (1610). This identification enables
24 the origin web-server to determine the key Kclient (see
25 Fig. 15). The semi-trusted web-server receives the
26 secure content for the client from the origin
27 web-server and serves the content to the client (1615).
28 Whether secure content can be cached or not depends on
29 the content itself and is determined by the origin
30 web-server. The procedure serving the secure content
31 part of a request ends with 1620.

1 Fig. 17 shows a flowchart that illustrates the actions
2 taken by the client when receiving secure content
3 (1700). First, the client's browser will check whether
4 it has a registered program for the type of content it
5 received (1705). If the client receives conventional
6 content then it processes the data with the existing
7 content handlers (1710) and returns (1715). If the
8 client receives data tagged as secure content, the
9 browser will automatically look for the respective
10 registered program to handle this content (1720). The
11 client recognizes secure content by its special content
12 tag (see Fig. 18). If the browser does not have
13 registered a handler for secure content, then the user
14 is prompted and must go through the installation
15 procedure. The client-side handler for secure content
16 is installed via a secure connection from the origin
17 web-server; throughout this process (1725), the secure
18 key Kclient is installed in the client program as well.
19 Kclient is known only to the client-side program and
20 the content provider, e.g., a cgi script or servlet in
21 the origin web-server. Next, the client program as part
22 of the web-browser is triggered to handle the secure
23 content-type. The client-side program extracts the
24 protected key, Kp, attached to the secure content
25 (1730) and computes the decryption key Kcontent for the
26 secure content by decrypting the key Kp with its own
27 secret key Kclient (1735). The resulting key Kcontent
28 is used to decrypt the secure content and check the
29 integrity of the secure content (1740). The resulting
30 cleartext can be handled as usual by the web-browser or

1 can be presented by the client-side program in a way
2 that shows users which part of a page is secure content
3 and which part is not (1745).

4 Fig. 18 illustrates the new content-type that is
5 attached to secure content (1800). A new tag (1805)
6 triggers the client's web-browser to activate the
7 client-side program (installed as described in Fig.
8 15). The protected key section (1810) comprises the
9 key, protected by Kclient, to decrypt the secure
10 content. The tag is followed by the encrypted secure
11 content (1815). One possibility to implement the secure
12 content tag (1805) is to use a special file extension
13 (e.g., ".sec"), which is handled by a new content
14 handler. The file itself contains the protected content
15 key (1810) and the secure content (1815). Another
16 possibility is to define a new HTML-Tag for secure
17 content, which implements the secure content tag (1805)
18 and a pointer to the file including the protected
19 content key (1810) and the secure content (1815).

20 It is noted that the foregoing has outlined some of the
21 more pertinent objects and embodiments of the present
22 invention. This invention may be used for many
23 applications. Thus, although the description is made
24 for particular arrangements and methods, the intent and
25 concept of the invention is suitable and applicable to
26 other arrangements and applications. It will be clear
27 to those skilled in the art that modifications to the
28 disclosed embodiments can be effected without departing
29 from the spirit and scope of the invention. The

1 described embodiments ought to be construed to be
2 merely illustrative of some of the more prominent
3 features and applications of the invention. Other
4 beneficial results can be realized by applying the
5 disclosed invention in a different manner or modifying
6 the invention in ways known to those familiar with the
7 art.

8 The present invention can be realized in hardware,
9 software, or a combination of hardware and software. A
10 visualization tool according to the present invention
11 can be realized in a centralized fashion in one
12 computer system, or in a distributed fashion where
13 different elements are spread across several
14 interconnected computer systems. Any kind of computer
15 system - or other apparatus adapted for carrying out
16 the methods described herein - is suitable. A typical
17 combination of hardware and software could be a general
18 purpose computer system with a computer program that,
19 when being loaded and executed, controls the computer
20 system such that it carries out the methods described
21 herein. The present invention can also be embedded in a
22 computer program product, which comprises all the
23 features enabling the implementation of the methods
24 described herein, and which - when loaded in a computer
25 system - is able to carry out these methods.

26 Computer program means or computer program in the
27 present context mean any expression, in any language,
28 code or notation, of a set of instructions intended to
29 cause a system having an information processing
30 capability to perform a particular function either

